Fast S-box Substitution Instructions and their Hardware Implementation for Accelerating Symmetric Cryptographic Processing

DUAN Cheng-hua, JIANG Jun, WANG Xing-ming, XU Wen-yuan

School of Information Science and Engineering
Graduate University of Chinese Academy of Sciences
Beijing, China
E-mail: cduan@gucas.ac.cn

Abstract—In popular symmetric ciphers, S-box substitution is the core operation that dominates executions of cryptographic algorithms. In this paper, a method of application-specific instruction-set extension is used for accelerating the key operation in symmetric cryptography. Two instructions for S-box access are designed by constructing a novel flexible on-chip parallel substitution box unit that consists of multiple lookup tables and a post-processing module. The box unit is integrated into the 32-bit configurable Leon2 processor. Configuration of Leon2 is presented. Implementing this extended processor core on Virtex-II XC2V3000 FPGA shows that the parallel substitution box unit uses very small amount of hardware resources (1KB of memory and some logic circuits). Evaluation of the performance of S-box access instructions for AES is conducted according to Amdahl Law, and the results show that overall speedup of greater than 2 can be achieved. Benefits for other symmetric ciphers using S-box substitution as their core operation are accordingly expected.

Keywords—Symmetric Cryptography; S-box Substitution; Lookup Table (LUT); Instruction Set Extension.

I. INTRODUCTION

The increasing need for secure communication and data encryption requires more information systems to process the workloads of cryptographic algorithms [1,2,3]. Today, cryptographic processing is widely used in embedded systems such as smart card, sensor nodes and mobile phones, etc.. However, the commonly-used pure software implementation of these cryptographic algorithms in embedded systems provides only a relatively low processing speed. Thus, hardware solutions are usually used in applications where high throughput cryptographic processing is needed [4]. But the hardware solutions still have drawbacks, such as lacking of function flexibility, high cost and long design cycle.

An alternative is the integrating of application-specific instruction into general purpose processor in order to better support cryptographic computations. Using this approach, cryptographic algorithms can be implemented efficiently due to the benefits of combining both hardware and software solutions. In symmetric-key cryptographic algorithms, S-box substitution operation is frequently used and costly [5,6,7]. S-box itself is the kernel of most symmetric ciphers, and the substitution operation is commonly realized through lookup table. Therefore, accelerating table lookups becomes an important issue.

Burke *et al.*[7] developed an S-box instruction to compute memory address and load data using this instruction, thus speeding up memory lookups. Compared to software lookup table, hardware table is more suitable for high speed implementation. By applying this approach, Tillich *et al.*[5] introduced a on-chip S-box table for AES substitutions. In order to support different symmetric algorithms, rewritable memory for storing S-box data is needed. Fiskiran *et al.*[6] take advantage of scratch-pad memory to construct a parallel table lookup unit for S-box substitution. With such on-chip lookup table, main memory access is avoided, which lowers sub-stitution cost efficiently. Therefore, it is attractive to took advantage of hardware lookup tables as an effective way for accelerating S-box substitution. However, lookup tables often occupy large memory resources. For instance, lookup tables introduced in [6] occupy a memory space of 4KB-16KB. Besides, effective matching of S-box structure and cipher algorithms remains an open problem.

In this paper, a novel parallel S-box table unit is constructed and used as the functional module of the integer unit of a processor, and two S-box instructions are developed for direct substitution operations. This parallel S-box table unit consists of four lookup tables and a post-processing unit, and characterizes as follows: it (1) occupies a small amount of hardware resources with simple structure; (2) can perform fast S-box substitution flexibly, and thus improves the efficiency of algorithms implementation; and (3) greatly lowers cost of the communication between hardware accelerator and main processor. In this paper, two instructions for S-box access are designed to support these operations as an extension of SPARC V8 architecture. Hardware functional unit that implements the proposed instruction set architecture (ISA) extensions is presented. This unit is integrated into the core of Leon2, being as a base machine, for cryptographic processing, leading to an extended processor core which is thereafter implemented on Xilinx Virtex-II XC2V3000 FPGA. The HDL model of the extended machine is synthesized to get gate level net-lists. Compared to the base machine, the extended one uses only a small amount of memory (1KB) and additional logic circuits for accelerating symmetric ciphers that employ S-box substitution without degrading the whole system.

The rest of the paper is organized as follows. In section II we discuss the S-box substitution operation in symmetric

ciphers. Section III develops the on-chip parallel S-box table unit and defines the corresponding S-box instructions. In section IV we describe our approach of integrating the S-box table unit into Leon2 processor core, and implement the extended machine on a Xilinx Virtex-II FPGA. The hardware cost of S-box unit is analyzed and the usage of S-box instructions is presented. Conclusions are drawn in section V.

II. S-BOX SUBSTITUTION IN SYMMETRIC-KEY ALGORITHMS

The operation of S-box substitution generates output vector using input vector through a well constructed S-box. Any changes to the input vector results in random-looking changes to the output, and their relationship is non-linear and difficult to approximate with linear functions [1]. S-box is the kernel of symmetric ciphers. It is characterized by its size $n \times m$, where n is the width of input and m the width of output. Commonly used symmetric ciphers are listed in Table I.

TABLE I. S-BOX IN SYMMETRIC CIPHERS

Algorithm	No. of boxes	Size
DES ^[8]	8	6×4
3DES ^[9]	8	6×4
AES ^[10]	1	8×8
Twofish ^[11]	4	8×8
RC4 ^[2]	4	8×8

Generally, larger S-boxes are more resistant to differential and linear cryptanalysis. On the other hand, the larger the S-box is, the more difficult it is to design properly [1]. As to the ciphers listed in Table I, the largest input vector width n is 8 and output vector width m is 8 too. So we can use a moderate sized scratch memory (8×8 for instance) to store S-box data.

III. PARALLEL S-BOX UNIT AND S-BOX INSTRUCTIONS

A. Design of Parallel S-box Unit

Figure 1. Block diagram of a parallel S-box unit

Considering the size and number of substitution boxes, we construct an 8×32 S-box unit. S-box substitution can be parallelized within one round, and the parallelism is only restricted to hardware resources. The 8×32 S-box unit is divided into four parallel banks with size of 8×8 for each one. Thus for DES and 3DES, two substitution boxes can be stored in each bank, occupying 64 upper entries. For AES, four-way parallel access can be achieved when taking the advantage of four identical substitution boxes. Twofish needs to use all the four banks and RC4 has both read table and write table operations. Therefore, multiple banks can be used to avoid structure hazard.

The parallel substitution box unit consists of two parts: memory unit and post-processing unit, as shown in Figure 1. The memory unit is divided into four identical SRAM banks, where A3-A0 are address signals of the four banks. The address width of each SRAM block is 8 bits and its data width is 8 bits too. The post-processing unit is a three-stage multiplexer network. The four-way data outputs from the memory go through this network to realize byte-wise permutation. The first stage selects data, and the later two constitute a logarithmic shifter. D3-D0 are post-processed data outputs and as feedbacks into the first stage of the multiplexer network. Therefore, this parallel substitution box unit can realize arbitrary bytes substitution. Its structure is simple and its delay is composed of only two parts: memory access delay and three-stage multiplexer network delay, which produce a small overall delay.

B. S-box Substitution Instructions

The S-box substitution instructions are *sboxrd* and *sboxwr*. Figure 2 and Figure 3 illustrate the functionality of *sboxrd* and *sboxwr* respectively. Their instruction formats abide by SPARC V8 specification [12] and are encoded using its format 3 class.

Sboxrd instruction is defined as

sboxrd *regrs1*, *imm*, *regrd*.

This instruction performs arbitrary byte-wise substitution. It selects one byte or up to four bytes of source operand *regrs1* depending on source operand *imm*, and performs substitutions using S-box or inverse S-box. At the same time, the output data is rotated byte-wisely. The result is written to *regrd*. Because source operand 2 is always immediate, the second read port of register file is used for accessing *regrd*. This instruction execution takes one single clock cycle.

Sboxwr instruction is defined as

sboxwr *regrs1*, *reg* or *imm*, *regrd*

This instruction supports four-way write operation to the S-box unit. It's used for initialization and reload, as well as write operation in RC4. The least significant eight bits of *reg* or *imm* is used as the memory write address. And the four bytes in *regrs1* are also written into *regrd*.

Figure 2. Functionality of sbxord instruction

Figure 3. Functionality of sbxwr instruction

IV. HARDWARE IMPLEMENTATION

A. Leon2 processor core

Leon2 is a SPARC V8 compatible processor core designed for embedded applications. It is developed and maintained by Gaisler Research [13]. Its non-fault tolerant edition has been released publicly. Figure 4 shows its structure sketch. It consists primarily of the following five parts: (1) integer unit, (2) floating-point unit and coprocessor, (3) cache subsystem, (4) memory management unit, and (5) system interface and peripheral equipment.

Figure 4. The structure sketch of Leon2 system

Due to its parameterized design adopted, the HDL model of Leon2 is endowed with excellent configurability. The configuration of Leon2 is listed below:

a. IU configuration:

The configurable elements are mainly the number of register windows (NWindows) and units for multiplication and division. Any number of register windows varying from 2 to 32 is feasible in SPARC. To achieve multiplication operation, shift-and-add or hardware multipliers with different bit-widths could be selected. Presently for division operation, only radix-2 divider could be used. In our configuration, the number of register widows is set to be 8, the hardware multiplier is chosen as 16×16 with 4 beats of delay, and the dividers are spared.

b. FPU configuration

For FPU configuration, we may choose among different FPU cores and interfaces. But in our current configuration, considering features of symmetric-key cryptographic algorithms, FPU is not used.

c. Cache configuration

Cache configurable properties include cache capacity, block size, set associative number and substitution algorithm. Instruction cache and data cache can be configured separately. In our configuration, instruction cache and data cache are all configured as 8KB respectively and direct mapping and random substitution algorithms are chosen.

We use Leon2 soft core as the basis and integrate the new hardware unit into the processor's integer unit. The extended integer unit is shown in Figure 5. The parallel substitution box unit acts as a functional module embedded into the execution stage of the 5-stage pipelined integer unit. Thus it can access all the resources of the processor and lower the cost of the communication between hardware accelerator and main processor.

Figure 5. Block diagram of the extended integer unit

B. Implementation on FPGA

In order to evaluate the hardware resource usage of the functional unit, we implement the extended processor core on Xilinx Virtex-II FPGA XC2V3000-FG676-6[14]. The configuration of register window number and the Harvard-style cache is set as stated before. For comparison purpose, the base processor core and the extended one are implemented on the same platform. Hardware resource usage is listed in table II. Compared to the base machine, the extended machine uses extra 1KB memory and four BRAMs on chip.

TABLE II. RESOURCE USAGE OF S-BOX

	Logic (No. of slices)	Memory (No. of BRAM)
Base machine	5148	20
Extended machine	5251	24
Increase	2%	20%

Post-PAR timing analysis of the base and extended machines shows that their system clock frequency retains 60MHz, which implies that the functional unit implemented on FPGA does not decrease the whole system speed.

C. The Usage of Sbox Instructions

We take AES as an example to illustrate how to use S-box substitution instructions.

In order to improve the implementation efficiency of AES on today's 32-bit processors, we can use a larger T-box derived from the original AES S-box [15]. The size of the T-box is 8×32 . Each substitution within an AES round requires four instructions: (1) a right-shift instruction to move the index byte to the rightmost byte of a temporary register, (2) a left-shift instruction to scale the index by 4x, which is needed to read 32-bit-wide tables, (3) an add instruction to add the scaled index to the table base address, and (4) a load instruction to get the data from memory. The SPARC assembly is shown as follows:

```
!SPARC assembly to read a (2^8)*32 table
using
!B2 of %10 as index. Let the 32-Bit
register %10
!contain 4 index bytes [B3 B2 B1 B0],
while %11
!contains the table base address.
SRL %10, 16, %12 !bring index byte
rightward
AND %12, 255, %12 !mask the upper 24 bits
SLL %12, 2, %12 !scale index by 4x
LD [%13+%11], %13 !load data
```

We assume the cache always hits, and then a 32-bit word substitution will cost 15 cycles. As for the case with a large amount of data involved, the initialization cost is minimal, thus a single *sboxrd* instruction can accomplish the word substitution: *SBOXRD %10, 15, %13*. Consequently, the fractional speedup compared to the pure software implementation is 15/4. Typically, substitution operation costs 70%

of the running time of AES on RISC processors. So according

to Amdahl's law, the overall speedup is $\frac{1}{0.3 + 0.7 \times \frac{4}{15}} = 2.05$.

V. CONCLUSION

Applying the approach of instruction set extension, we present hardware support at the instruction set architecture level for S-box substitution operation. We construct an on-chip parallel substitution box unit and integrate it into the base processor core to form an extended core with cryptographic accelerator. Our accelerator occupies only a small amount of hardware resources and does not lower system clock frequency. Using the instructions we define on the SPARC V8 instruction set architecture, S-box substitution can be speeded up significantly. Because using S-box instructions can achieve high fractional speedup and substitution operation costs a large proportion of symmetric cipher's running time, benefits for other symmetric ciphers using such S-box substitution as their core operation are accordingly expected.

REFERENCES

- [1] W. Stallings, *Cryptography and Network Security: Principles and Practices*. Prentice Hall, 2006.
- [2] B. Schneier, *Applied Cryptography*, New York: Wiley, 1996.
- [3] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC, 1996.
- [4] Alireza Hodjat, Ingrid Verbauwhede. "High-throughput Programmable Cryptocoprocessor," *IEEE Micro*, May-June 2004.
- [5] Stefan Tillich and Johann Grossschadl, "Instruction Set Extensions for Efficient AES Implementation on 32-bit Processors," *CHES 2006*, LNCS 4249, 2006, pp. 270-284.
- [6] A. Murat Fiskiran and Ruby B. Lee, "On-Chip Lookup Tables for Fast Symmetric-Key Encryption," *Proceedings of IEEE 16th International Conference on Application-Specific Systems, Architectures, and Processors*, Jul. 2005, pp. 356-363.
- [7] Jerome Burke, John McDonald, and Todd Austin, "Architectural Support for Fast Symmetric-Key Cryptography," *Proceedings of ASPLOS*, 2000.
- [8] National Institute of Standards and Technology (NIST), "Data Encryption Standard (DES)," *FIPS Publication 46-3*, Oct. 1999.
- [9] National Institute of Standards and Technology (NIST), "Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher," *NIST Special Publication 800-67*, May 2004.
- [10] National Institute of Standards and Technology (NIST), "Advanced Encryption Standard (AES)," *FIPS Publication 197*, Nov. 2001.
- [11] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall and N. Ferguson, "Twofish: A 128-Bit Block Cipher," <http://www.schneier.com.sixxs.org/papertwofishpaper.html>, Dec. 3, 2007.
- [12] SPARC International, *The SPARC Architecture Manual*, Version 8, Revision SAV080SI9308, 1992.
- [13] Gaisler Research, *LEON2 Processor Users Manual*, XST Edition, Version 1.0.30, 2005.
- [14] Xilinx, "Virtex-II Platform FPGAs: Complete Data Sheet," <http://www.xilinx.com>, Dec. 2007.
- [15] B. Gladman, "AES and Combined Encryption Authentication Modes," <http://fp.gladman.plus.com/AES/index.htm>, Dec. 2007.